CLAIMS

1. A method comprising:

loading one or more source processing chains to support execution of a development project; and

determining whether each of the one or more processing chains will be subsequently required during execution of this or another development project and, if so, caching those filter chains which will be subsequently required.

2. A method according to claim 1, wherein loading of the one or more processing chains comprises:

identifying which source(s) will be required to support execution of the next M seconds of the development project;

searching one or more cache(s) to determine whether the source processing chain(s) associated with the source(s) are available within the one or more cache(s); and

retrieving the one or more processing chains from a memory location denoted by an associated one or more pointers in the cache for integration with the development project.

3. A method according to claim 2, further comprising:

determining whether processing chain(s) retrieved from the cache(s) satisfy processing requirements of the development project; and

modifying one or more objects of one or more of the processing chain(s) retrieved from the cache(s) that do not satisfy the processing requirements of the development project.

4. A method according to claim 3, wherein modifying one or more objects may comprise one or more of adding processing objects to the processing chain(s), removing one or more processing objects from the processing chain(s), or changing one or more operating attributes of one or more processing objects within the processing chain(s).

5. A method according to claim 2, wherein M is at least as long as necessary to construct a processing chain.

6. A method according to claim 1, wherein determining whether a processing chain will subsequently be required comprises:

determining whether any future calls to a source coupled to the processing chain exist within this development project; and

determining whether any future calls to a source coupled to the processing chain may be received during execution of future development projects.

7. A method according to claim 6, wherein it is assumed that each processing chain may well be required to support future execution of this or a future development project.

8. A method according to claim 1, wherein caching the processing chain comprises:

assigning the processing chain a unique identifier; and

storing the unique identifier along with a pointer to a memory location occupied by the processing chain in a cache.

9. A method according to claim 8, wherein the unique identifier is one or more of a source file handle, a source file name, a random numeric identifier uniquely assigned to the processing chain, a graphical icon, an alphanumeric character, and the like.

10. A storage medium comprising a plurality of executable instructions which, when executed, implement a method according to claim 1.

11. A computing system comprising:

a storage medium having stored therein a plurality of executable instructions; and

an execution unit, coupled to the storage medium, to execute at least a subset of the plurality of executable instructions to implement a method according to claim 1.

12. A method comprising:

generating a source chain for use in a development project; and

caching the source chain when it is not currently required in the development project.

13. A method according to claim 12, wherein the processing chain is cached only if it will subsequently be required in the development project, or a future development project.

14. A method according to claim 12, wherein caching the source chain comprises:

generating an identifier which is uniquely assigned to the processing chain; and

storing the identifier along with a pointer to memory occupied by the processing chain in a cache of processing chain pointers.

15. A method according to claim 14, wherein the identifier is one or more of a source file handle, a file name, an icon, a randomly generated number uniquely assigned to the processing chain, an alphanumeric identifier, and the like.

16. A method according to claim 12, further comprising:

identifying a need for a source processing chain; and

retrieving a suitable processing chain from a cache of such processing chains.

17. A method according to claim 16, further comprising:

integrating the retrieved processing chain into the development project.

18. A method according to claim 16, further comprising:

modifying one or more attributes of the retrieved processing chain before integration into the development project.

19. A method according to claim 18, wherein modification to the retrieved processing chain may include one or more of adding processing objects to the processing chain, removing processing objects from the processing chain, altering one or more processing characteristics of one or more processing objects of the processing chain, and the like.

20. A storage medium comprising a plurality of executable instructions which, when executed, implement a method according to claim 12.

21. A computing system comprising:

a storage medium having stored therein a plurality of executable instructions; and

an execution unit, coupled to the storage medium, to execute at least a plurality of the instructions to implement a method according to claim 12.

22. A system comprising:

a plurality of sources; and

an interface, selectively coupled to the plurality of sources, to generate and implement a development project of processing chains, wherein the interface loads a processing chain for each of the plurality of media sources at a point during the

execution of the project when the chain is required, and wherein the interface is configured to unload at least a subset of the chains when they are not required.

23. A system according to claim 22, wherein the interface only loads those processing chains required during the next M seconds of project execution, and if a current chain-count does not exceed a threshold, T.

24. A system according to claim 23, wherein M is less than a time required to load a processing chain.

25. A system according to claim 23, wherein if the currently loaded chain-count has reached a threshold, T, the interface identifies one or more currently loaded chains that can be unloaded.

26. A system according to claim 25, wherein the interface identifies one or more currently loaded chains that will not be used during the next N seconds to unload.

27. A system according to claim 25, wherein the interface determines whether the identified one or more chains will be required during subsequent execution of the project, or in a future project and, if so, caches the identified chain(s).

28. A system according to claim 27, wherein the interface assigns a unique identifier to processing chains to be cached, and stores the unique identifier along with a pointer to memory wherein the processing chain resides in a processing chain cache.

29. A system according to claim 22, wherein the interface removes the identified chains from the active project and caches the removed chains.

30. A system according to claim 22, wherein the interface loads processing chains by first searching a cache of processing chains for a suitable match.

31. A system according to claim 30, wherein if the interface identifies a suitable match, the processing chain is retrieved from memory for integration within the processing project.

32. A system according to claim 31, wherein the interface modifies one or more attributes of the retrieved processing chain before integration within the processing project.

33. A system according to claim 32, wherein modifying the processing chain, the interface performs one or more of adding one or more processing objects to the processing chain, removing one or more processing objects from the processing chain, modifying one or more processing characteristics of one or more processing objects within the processing chain.